ASG: Automatic schematic generator

Yeu-Shen Jehng, Liang-Gee Chen and Tai-Ming Parng

Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan 10764

Received 16 March 1990

Abstract. In this paper a heuristic placement and routing method with a value propagation approach, the bubbling technique, is presented to do automatic schematic generation to match human esthetics. Algorithms embedded in the method are derived to link the schematic placement and routing in human habits. Meanwhile, the left edge algorithm is modified with channel routing aspect to achieve the compact channel size and regular channel routing. The primary considerations include circuit levelling, feedback loop detection, signal flow resolution, feedback and fanout signal routing, channel size estimation, and track assignment. The implemented system, automatic schematic generator (ASG), accepts the results of high level syntheses, schedules the schematic placement and routing, and finally, commits the browsing to the schematic editor to support friendly user interfaces. The tested examples show that the proposed system can really achieve good results which are near human design.

Keywords. User interface, automatic schematic generation, value propagation, heuristic placement and routing.

1. Introduction

As the rapid progress of VLSI process techniques, the design circuit becomes more complex. It leads to more design data handled by CAD design automation systems and more graphic overheads to enhance their man–machine interfaces. Since a schematic diagram is one of the most comprehensive design representation for human designers, it is important to have a tool which can automatically generate schematic diagrams to serve as a pipe for the communication between a design automation system and human designers. However, schematic drawing is

Elsevier
INTEGRATION, the VLSI journal 11 (1991) 11–27

0167-9260/91/$03.50

such an artwork that everyone may have different views. Therefore, schematic diagrams must be regular, symmetrical, functionally readable, generally understandable and esthetically pleasing to human designers when they are utilized to browse the primary functions in circuitry such that they can satisfy as many circuit designers as possible.

To meet the above requirements, the schematic drawing with component placement and net routing must be considered in human aspects. However, it is not easy to derive the schematic placement and routing algorithms with human esthetic considerations. There have been many researches who attempted to provide this feature. Ahlstrom [6] supported the Heuristically Augmented Layout (HAL) prototype designed by a rule-based expert system. This serves as an intelligent interface for a front–end digital system design tool to produce a schematic drawing from a hardware description file. But the knowledge inferences will slow down the overall processing in a general purpose computer. Kumar [1] and Stok [4] described a module networking algorithm and emphasized

Yeu-Shen Jehng was born in Taipei, Taiwan, R.O.C., in 1963. He received the B.S. in Electrical Engineering from the University of Chinese Culture, and M.S. degrees in Electrical Engineering from National Taiwan University, in 1987 and 1989, respectively. He is currently working towards the Ph.D. degree in department of Electrical Engineering at National Taiwan University. His research interests include VLSI CAD frameworks, DSP architecture design and video coding for communications.

Liang-Gee Chen was born in Yun-Lin, Taiwan, R.O.C., in 1956. He received the BS, MS, and Ph.D degrees in Electrical Engineering from National Cheng Kung University, in 1979, 1981, and 1986, respectively.

He was an Instructor (from 1981 to 1985), and an Associate Professor (from 1986–1988) in the Department of Electrical Engineering, National Cheng Kung University. In the military survice during 1987 and 1988, he was an Associate Professor in the Institute of Resource Management, Defense Management College. Currently, he is an Associate Professor in the Department of Electrical Engineering at National Taiwan University. His current research interests are DSP architecture design, Silicon Compilation, and Logic Synthesis.

Dr. Chen is a member of IEEE, and the Association for Computing Machinery. He is also a member of the honor society Phi Tan Phi.

Tai-Ming Parng is a professor in the Electrical Engineering Department at National Taiwan University (Taiwan). He received his BS, MS, and Ph.D. degrees from the same university in 1971, 1973, and 1981 respectively. He has been a visiting scholar to IBM T.J. Watson Research Center (1978–1979), Carnegie-Mellon University (1983–1984), and UC Berkeley (1988–1989). His current research interests are in knowledge-based ASIC specification and design, framework for system-level design automation, as well as VLSI architectures for parallel processing. He is a member of IEEE.

on schematic placement and routing. The drawback is that it is not easy to blend heuristics into algorithms to recognize all of the circuit patterns. Majewski [5] took the advantage of an adjacency matrix to process constructive initial placement and routing to draft schematic diagrams. Chun [7] extracted net lists from VHDL and applied an incidence matrix to detect the feedback loops. However, the latter two systems which focused their processing on matrix operations may slow down the system speed when large circuits are processed.

Based on the discussions mentioned above, the major problems in the previous researches can be summarized as follows:

(1) Lack of global consideration of full circuit interconnection and distribution.
(2) The emphases on routibility are similar to layout routing but lose the feeling of schematic esthetics.
(3) Lack of considering the drawing habit of human designers.
(4) The massive matrix operations may slow down the system speed when large amounts of data are processed.

To solve the above problems, schematic placement must be considered to follow the signal flows of circuit and routing must be specially concerned for the feedback and fanout interconnections rather than just a blind search. There are several basic considerations described as follows:

(1) For the purpose of reducing the complexity of the circuit, arbitrarily shaped functional modules with terminals all around can be used hierarchically.
(2) In human habit, the signal flows of circuits are always scheduled to go from left to right. To match this habit, the placement should also be considered to follow the signal flows from left to right.
(3) The feedback loops in circuits are always grouped together, especially for the latch loops in sequential circuits. These groups will be processed individually. It will expose the circuit functions to enhance the readability of circuits.
(4) The fanout and feedback interconnections must be extracted to schedule them regularly, straightly, and evidently to express the characteristics in circuitry.
(5) The pin assignment can exchange pins to reduce the number of interconnection crossovers. In general, the feedback pins will be assigned to the boundary of the feedback side.
(6) The regularity, symmetry, and tidiness are more important than the minimization of crossovers, bends, and path length.
(7) To speed up the processing and for the convenience of algorithm traversal, it is better to represent the circuit in dynamic data structure.

Based on these considerations, we model the circuit as a directed graph in a generalized list representation and develop a heuristic placement and routing method with a value propagation approach called *the bubbling technique*. This method supports algorithms to link the schematic placement and routing in human aspects and emphasizes on the routing considerations especially for feedback and fanout signals, channel size estimation, and track assignment.

In this paper, Section 2 describes the environment of the generator and the internal data structure which was used to model the circuit. Section 3 describes

the heuristic placement and routing method in detail, and the implementation and some tested examples are analyzed in Section 4.

2. System description

Figure 1 describes the environment of the generator – ASG. The VLSI chip design may begin by entering the hardware description languages which include the functional and structural definitions of designed circuits. The high level synthesis tools accept the languages and allocate components by selecting designed modules in the design data base to generate the structural component (module) description form. Figure 2(a) illustrates the form corresponding to the example circuit shown in Fig. 2(b). It describes the modules to be used and the full interconnections between the modules. Also, it acts as a common entry of ASG and is portable between CAD systems to solve the incompatibility of design data bases. ASG is invoked by the schematic editor to accept the structural component description form and access information (graphic shape, pin count, etc.) about the components from design data base to detailedly schedule their placement and net routing. At last, ASG controls the schematic editor to browse the final arrangements of the circuit and the designer may interactively revise the circuit or commit the final design to the design data base.

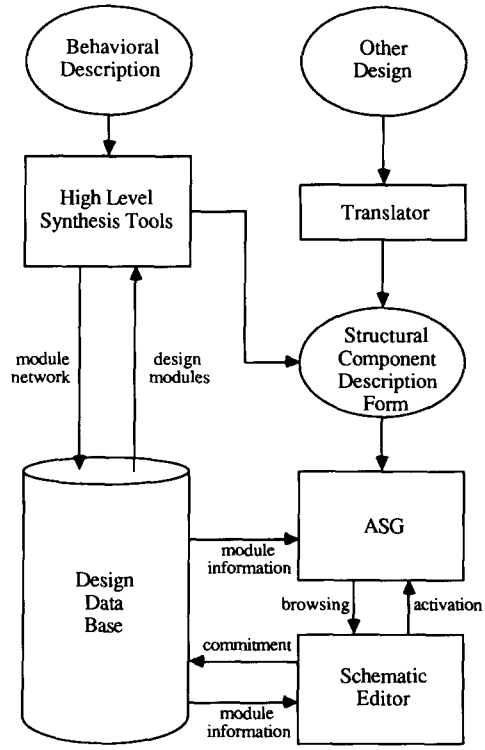

Fig. 1. The environment of the generator.

(a)

```
module  JK_MASTER_SLAVE_FF( in , o )
{
    input      in[3];
    output     o[2];
    internal   int[7];

    gate nd_3 { nand 3 }
    gate nd_2 { nand 2 }
    gate inv  {  not  }

    instance gate nd_3 { nd1 , nd2 }
    instance gate nd_2 { nd3 , nd4 , nd5 , nd6 , nd7 , nd8 }
    instance gate inv { inv9 }

    connect nd1 ( o[1] , in[0] , in[2] , int[0] );
    connect nd2 ( in[1] , in[2] , o[0] , int[3] );
    connect nd3 ( int[0] , int[4] , int[1] );
    connect nd4 ( int[1] , int[3] , int[4] );
    connect nd5 ( int[1] , int[6] , int[2] );
    connect nd6 ( int[4] , int[6] , int[5] );
    connect nd7 ( int[2] , o[1] , o[0] );
    connect nd8 ( o[0] , int[5] , o[1] );
    connect nd9 ( in[2] , int[6] );
}
```

(b)

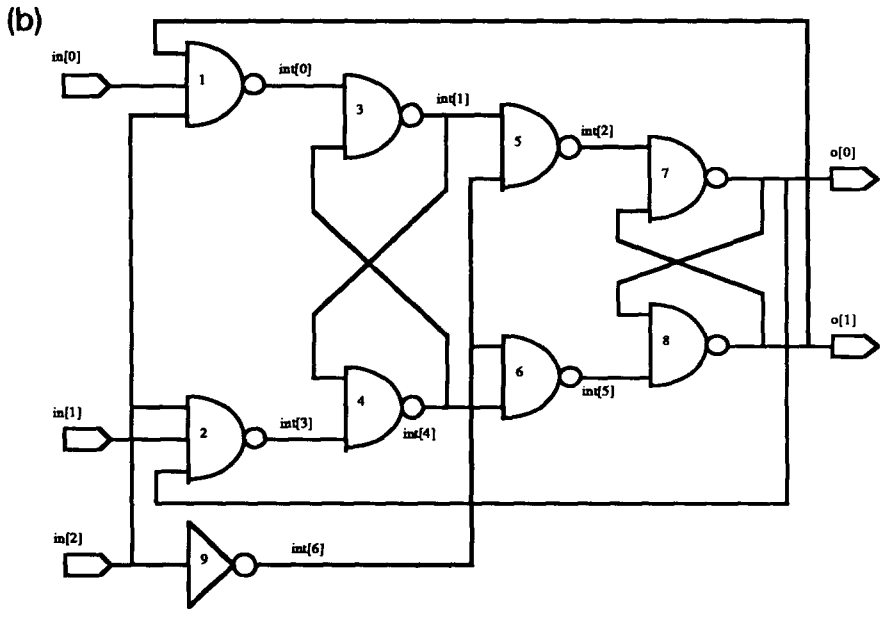

Fig. 2. Structural component description form: (a) structural component description form for the example circuit; (b) example circuit.

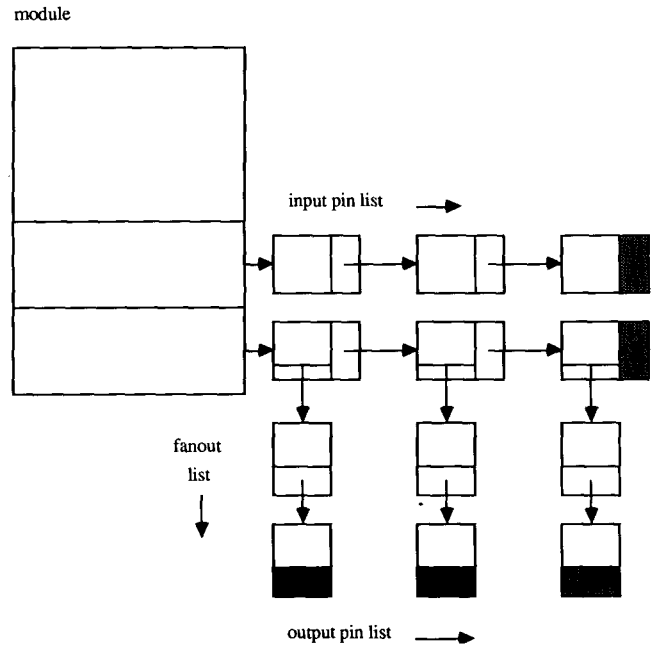

Fig. 3. Internal data structure – a generalized list.

In ASG processing, the circuit is modelled as a directed graph in a generalized list representation which is shown in Figure 3. For each input pin, output pin, and fanout of modules, there exists a list node to keep their relation between modules respectively. Therefore, all relations in circuitry can be constructed in the generalized list structure. It will both benefit the algorithm traversal either forward or backward and speed up the overall processing. In this paper, the signal flow of the circuit is always considered from left to right so the input and output ports are limited at the left and the right side of the circuit, respectively.

3. Heuristic placement and routing method

The heuristic placement and routing method can be divided into two phases. The first phase, called *the logical phase*, finds the relationships between modules to generate the initial placement. The second phase, called *the physical phase*, estimates the channel size around modules and assigns the routing signals to tracks in channels detailedly. To summarize the overall processing, the design flow is as follows:

Logical phase:

(1) extract the circuit connectivities from the structural component description form;

(2) partition the circuit into levels from left to right with respect to its signal flows:

(3) for each level, resolute the vertical precedence of modules;
(4) detect the fanout and feedback interconnections and make room for their routing in advance;
(5) pin assignment for each module.

Physical phase:

(6) select the most heavily connected level L and fix it in coordinates;
(7) include both of the adjacent levels into L and estimate their channel size around modules as well as exactly assign nets to tracks in channels;
(8) repeat (7) to expand L until all levels are included;
(9) follow the resultant schedule to do the physical placement and routing.

3.1. Logical phase

In this phase, the circuit will be considered first with respect to its signal flows in both the horizontal and vertical direction to construct the initial placement. Furthermore, the fanout and feedback interconnections can be extracted to make room for their routing. The purpose of considering this in advance is to render their routing more straight, regular, and evident without any obstacle. After the above processing, the pins of the module can be assigned by comparing the

(a)

```
module example( in, o )
{
    input       in[3];
    output      o[2];
    internal    int[4];

    instance    A, B, C, D, E;

    connect     A ( int[0], in[0], int[1] );
    connect     B ( in[2], int[2] );
    connect     C ( int[1], int[3] );
    connect     D ( in[1], int[2], o[1] );
    connect     E ( int[3], int[0], o[0] );
}
```

(b)

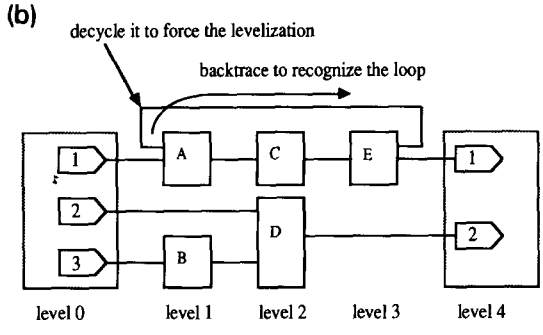

Fig. 4. Example for levelling algorithm: (a) example for circuit structural component description form; (b) circuit after levelling.

precedence of the modules connected to it. It will reduce the crossovers for further routing. The detailed steps can be described as follows:

(1) *Levelling*. Originally, the circuit may be described by using a structural component description form. An example is shown in Fig. 4(a). To begin with the processing, the connectivities must be firstly extracted from the description form and then the circuit can be modelled as a directed graph for the convenience of traversal. The levelling algorithm is derived to have the abilities of directed graph traversing, loop decycling, breadth first ordering, and module grouping for feedback loops. It unifies the signal flows of the circuit to go from left to right. By horizontally partitioning the circuit into sections, called the levels, with respect to its signal flows, the initial horizontal placement can be achieved. In the traversal, the feedback loops will be detected and decycled to continue the traversing, in the meantime, modules in feedback loops will be extracted and grouped together in the neighborhood. It will make the feedback loops more evident in the drawings. Figure 4(b) illustrates the circuit after levelling. There are two parts in the algorithm: PART I tries the best to accumulate the levels in the order of breadth first search. If there is no feedback in the directed graph, it need not execute PART II. Otherwise, whenever PART I can not continue traversing the directed graph because there is no zero-indegree module, PART II is executed to detect the feedback loop by backtracking and then return to continue the PART I processing. By interleaving the PART I and II processing, the circuit can be levelled to achieve the horizontal placement. The proposed algorithm can be described as:

Part I: levelling and feedback detection

(1) Insert the input ports IP into BFS-queue;
(2) Empty the feedback-queue;
(3) **repeat**
(4) Remove element A from BFS-queue;
(5) **if** (indegree of $A = 0$)
then
(6) **for** (each module P connected to A)
(7) Set the level of P next to A if P was not behind A originally;
(8) Decrement the indegree of P;
(9) Union P to BFS-queue;
end for
(10) **else** insert A to feedback-queue;
(11) **until** BFS-queue is empty;
(12) Remove element in feedback-queue which has zero indegree;
(13) **if** (feedback-queue is empty)
(14) **then** exit with success;
(15) **else** branch to **part II**;
end part I.

Part II: loop resolution and decycling

(1) Select the one A with the smallest level in feedback-queue;
(2) Backtrack from A to find a loop;
(3) **if** (there exists a loop)
then
Resolute the loop and group them together;
(5) Decrement the indegree of A;
(6) Insert A to BFS-queue;
(7) Remove A from feedback-queue if A has zero indegree;
(8) Branch to **part I (3)**;
else
(9) Remove A from feedback-queue;
(10) Branch to **part I (13)**;
end part II.

(2) *Bubbling resolution.* After levelling, the input ports are weighted with decreasing values, called the bubble values, in accordance with the precedence predefined by the designer as illustrated in Fig. 5(a). To weight all of the modules, the bubble values will be propagated through the circuit. Figure 5(b) depicts the propagation. Each module averages the values which came from the modules attached to its input pins and propagates it to the modules attached to its output pins. The propagation continues until the output ports. After propagation, the modules with the same level can be sorted in decreasing order of their bubble values to obtain the vertical precedence in each level. To explain the phenomenon of bubbling resolution, the modules with larger bubble values will go up, while the smaller ones go down. That is why it is called the bubble value. To evaluate the bubble values, there are some decisions described as follows:

(1) The values which came from the levels except for the immediate previous level will not be considered for evaluation;
(2) The values which came from fanout signals will not be considered for evaluation;
(3) If there is only one value to be considered, it is the bubble value.

The reason to make the above decisions is to prevent the fanout, feedback, and long path interconnections from confusing the local resolution in order to obtain the objective bubble values. However, the fanout, feedback, and long path interconnections can not be ignored and will be specially concerned in later processing. As the examples illustrate in Fig. 5(c), the number of signal crossovers is reduced and makes a clear drawing. The bubbling technique not only benefits the vertical resolution but also the fanout, feedback signal scheduling, and pin assignment. The later discussions will describe the reasons.

(3) *Fanout and feedback detection.* To generate a human like drawing, it is essential to control in detail the fanout and feedback routing in human style because both of them seriously reflect the human design characteristics in circuits. The general fanout and feedback routing style in human habits are depicted in Fig. 6. It can be viewed as a trunk with branches. In order to generate the human style routing, a trunk routing algorithm can be added to our bubbling technique.

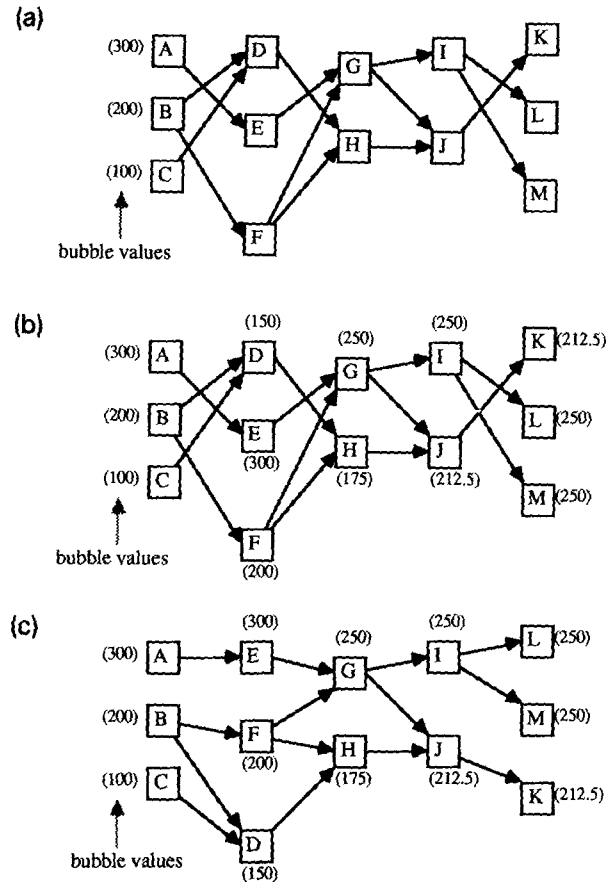

Fig. 5. Bubbling technique: (a) assign initial bubble values to input ports; (b) bubble value propagation; (c) sorting with bubble values in decreasing order for each level.

The procedures are denoted as follows:

(i) Regard output pin as the source and input as the destination. Their bubble values are inherited from their connected module respectively;
(ii) The trunk is presented straight in the direction of the destinations;
(iii) Each of the destinations projects perpendicularly to the trunk to create the branches; if many destinations project to the same perpendicular point on the trunk, they share the same branch;
(iv) Adjust the branches and trunk into the channels and make room for their routing.

To hold space for trunk routing, a so-called pseudo module can be inserted to occupy the holding space. The pseudo modules reserve space for trunk routing rather than for module placement. That is why it is called a pseudo module. As shown in Fig. 7(a), the fanout trunk goes through several levels. To make it go straight, the spaces which it passes must be reserved by pseudo modules. Thus, by comparing the bubble value of the trunk to those of the modules in each passed

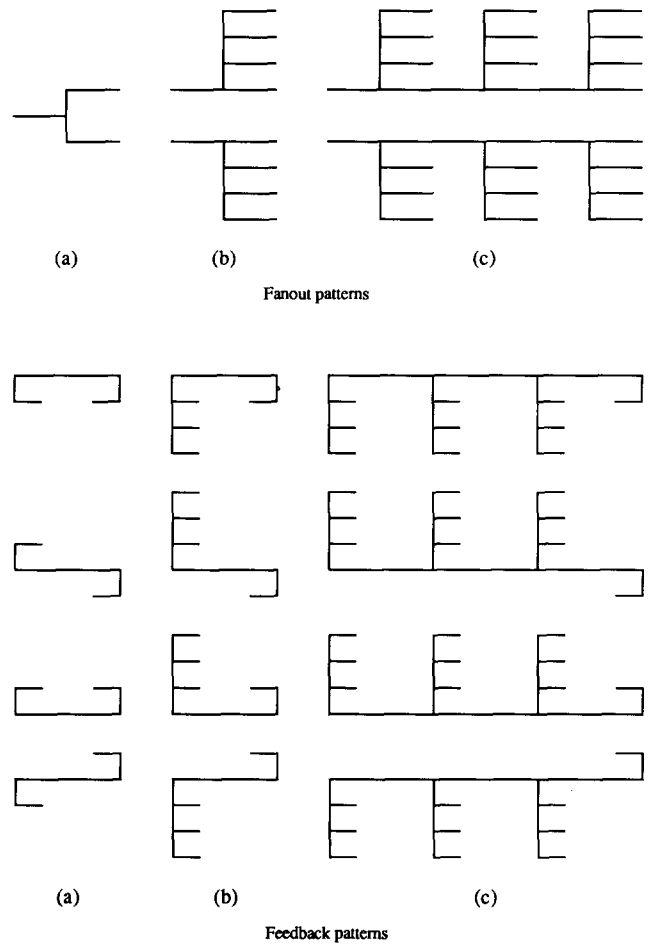

Fig. 6. General routing style for fanout and feedback.

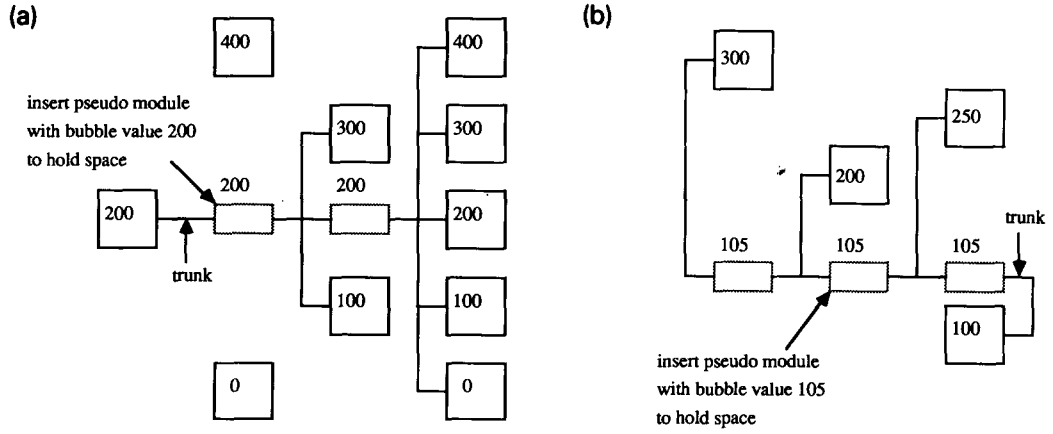

Fig. 7. Feedback and fanout considerations: (a) fanout trunk routing and pseudo module insertion; (b) feedback trunk routing and pseudo module insertion.

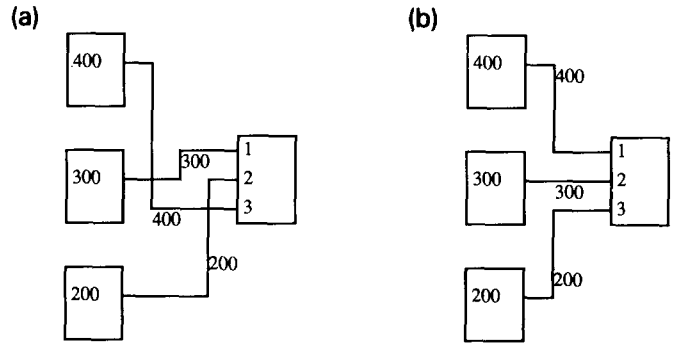

Fig. 8. Example for pin sorting with respect to bubble values: (a) before pin sorting; (b) after pin sorting.

level, the pseudo modules can be inserted to hold space for trunk routing appropriately. Figure 7(b) also illustrates the feedback trunk routing. The above processing will make the fanout and feedback routing more comprehensive and will expose the characteristics of human design in circuit to achieve a readable drawing.

(4) *Pin assignment.* After initial placement, the precedence between modules can be obtained. However, since the initial relative ordering of pins to a module is irrelevant from the current scheduling of initial placement, the pin assignment must be reconsidered with respect to the current scheduling to attain the esthetic goals of reducing the crossover of signal paths and increasing the symmetry. To consider the modules with exchangeable pins, the pin assignment can be done by in decending order sorting the bubble values of modules attached to their pins. From the sorted order, the pins can be assigned accordingly. Besides, for those pins that are attached to feedback signals, they are assigned to the boundary of modules. As shown in Fig. 8, the number of crossovers is greatly reduced and the drawing becomes more symmetrical and clearer.

From the above scheduling, the initial placement can be constructed, moreover, the fanout, feedback, and pin assignment can be considered in advance to profit the later routing. It is evident that the bubbling theory benefits most of the processing to achieve the heuristic placement and routing method.

3.2. Physical phase

The primary goal in this phase is to perform the detailed placement and routing. Often, the most heavily connected level plays an important role in a design and it can be viewed as the kernel of circuit. Thus, the most heavily connected level must first be selected and fixed to start the level expansion. Consequently, the other levels can easily be attached to it with little adjustment to achieve normal distribution in placement without the overhead of readjustment in levels. To begin with the expansion processing from the selected level, the adjacency levels will be included. Therefore, the channel size between them can

be estimated and the routing in channels can be assigned exactly to the tracks. The processing continues to the rest levels until the whole levels to be included. Thus, the overall scheduling of detailed placement and routing can be accomplished. Finally, the schematic editor will follows the resultant scheduling to generate the whole schematic diagrams and the designer can revise it interactively to commit the overall design. In the following, the detailed method will be described.

(1) *Level expansion from the most heaviest connection.* After the initial scheduling for placement and routing, the most heavily connected level will be selected to start the level expansion processing. The top module in the selected level will be

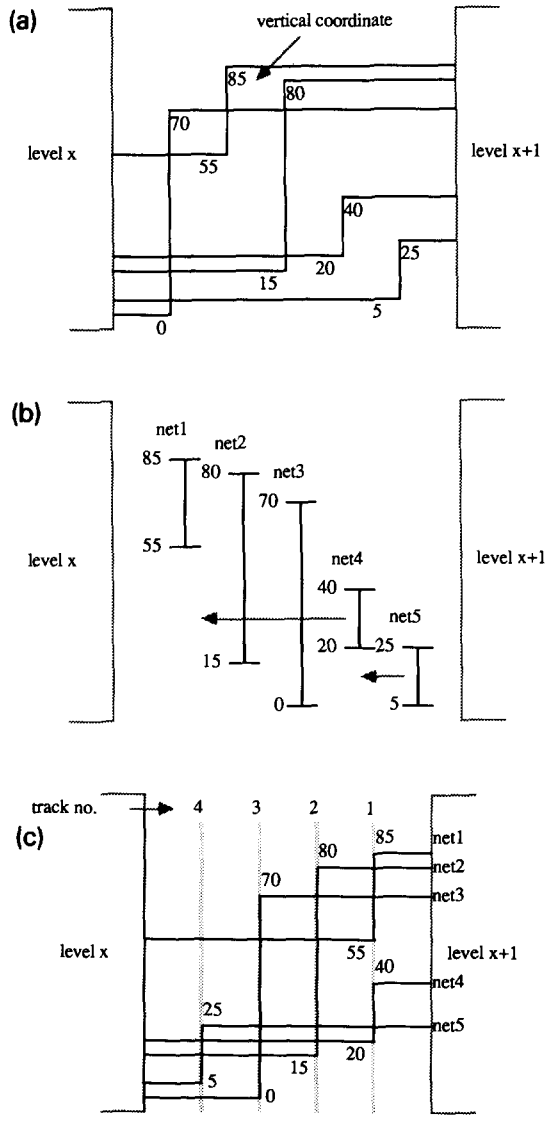

Fig. 9. Example for channel estimation and track assignment: (a) initial nets; (b) sorted nets; (c) after channel estimation and track assignment.

firstly fixed in coordinates. In the selected level, the size of intergaps between modules can be obtained by evaluating the space reserved by pseudo modules and the routing space for vertical pins. By continuously accumulating the intergaps, the other modules in the selected level can also be fixed in coordinates. Besides, if there exist latch loops in level, they must be grouped together to enhance the readability of drawings. After having fixed the selected level, the other levels can be attached to it accordingly with vertical adjustment to aim the output pin at the connected input pin in order to reduce the bended nets. By estimating the channel size between levels and assigning the nets to tracks in the channel, the expansion can be done level by level and the whole global scheduling is completed. The method to estimate channel size between levels and intergaps between vertical modules as well as the track assignment will be described later.

(2) *Channel size estimation and track assignment.* The left edge algorithm [2] can be modified with channel routing aspect (Fig. 9). To start the processing, the vertical segments of bended nets in the channel must first be collected to be sorted in decreasing order with respect to the position of top edge. From the sorted order, the segment with a small vertical coordinate is tested if it can be filled to the same track occupied by the segment with a large vertical coordinate without range conflict. If yes, it is assigned to the track. Otherwise, it tries the descending track in the same manner. The algorithm can be described as follows:

(1) Collect the vertical segments of bended nets;
(2) Sort the segments with respect to the position of top edge in decreasing order;
(3) From the sorted order, denote the nets to be NET_i; $1 \leqslant i \leqslant N$ where N is the total net number;
(4) Denote NET_i to be in track NET_{ik}; $1 \leqslant i \leqslant N$; $1 \leqslant k \leqslant T$ where T is the channel size;
(5) $\mathrm{NET}_{ik} \leftarrow 1$; for $1 \leqslant i \leqslant N$;
(6) **for** (i from 2 to N)
(7) 1: **for** (j from NET_{ik} to $i-1$)
(8) **if** (($\mathrm{NET}_{jk} = \mathrm{NET}_{ik}$) and (the top position of NET_i is within the range of NET_j))
then
$\mathrm{NET}_{ik} \leftarrow \mathrm{NET}_{ik} + 1$;
(10) **goto** 1;
end if
end for j
end for i
(11) Channel size $T \leftarrow \mathrm{MAX}\{\mathrm{NET}_{ik};\ 1 \leqslant i \leqslant N\}$;
(12) The NET_{ik} indicates the assigned track for NET_i; $1 \leqslant i \leqslant N$.

This algorithm not only finds the compact channel size but also assigns the nets to the exact tracks in decreasing order. The advantage of this application is that the scheduled circuit is tight enough and does not need the extra compaction process used in [1] and that the routing is tidy. This algorithm can not only be used for channel processing between levels but can also be used for the size

estimation of the gaps between vertical modules in each level. By cooperating channel processing with level expansion, the overall scheduling is thus accomplished.

(3) *Physical placement and routing.* To do the real placement and routing, the schematic editor follows the final scheduling to browse the overall schematic diagrams and takes over the control to continue the interactive processing.

4. Implementation and results

The ASG is written in C and works on a SUN 3/110 workstation. This system is linked with a database management system [3] to support the ability of a user-friendly interface and high level synthesis browsing. Figures 10–12 show the schematic diagrams automatically generated by ASG. Table 1 shows the statistical data and run time including the full graphic response time, number of modules, and connections. The characteristics of schematic diagrams generated by ASG can be analyzed as follows:

(i) From Figure 10, two latch loops can be detected and grouped together respectively. Both of the fanout and feedback signal paths follow the trunk routing algorithm. Consequently, the final drawing can be immediately identified as J K master slave flip-flop.

(ii) From Figure 11, the J K flip-flops with vertical pins can be processed and the feedback with fanout signal paths can be routed in human habits. The same as in (i), the final drawing is fully identical to that in databook.

(iii) From Figure 12, the more complex fanout signal paths are included. It can be seen that the routing in channels is very regular in decreasing order. It is profited from the channel size estimation and track assignment.

The experiments have shown that the drawings are very close to the human designs and have the characteristics of regularity, symmetry, functional readability, and artistry as expected. Moreover, the generation time is also endurable for the circuit designers.

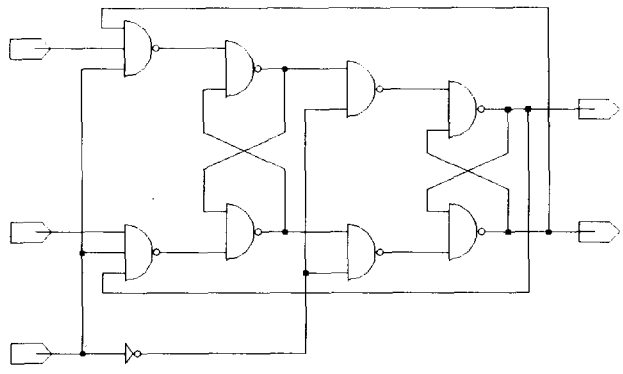

Fig. 10. J K master slave flip-flop circuit.

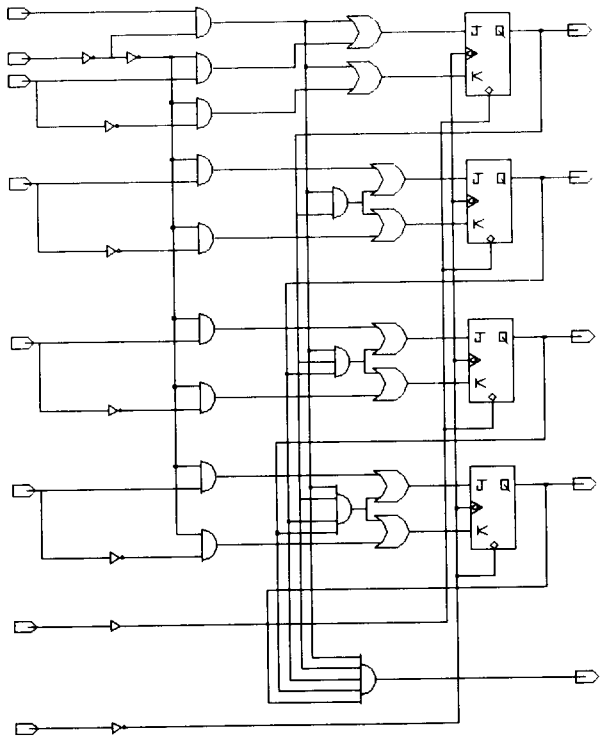

Fig. 11. Parallel load 4-bit binary counter.

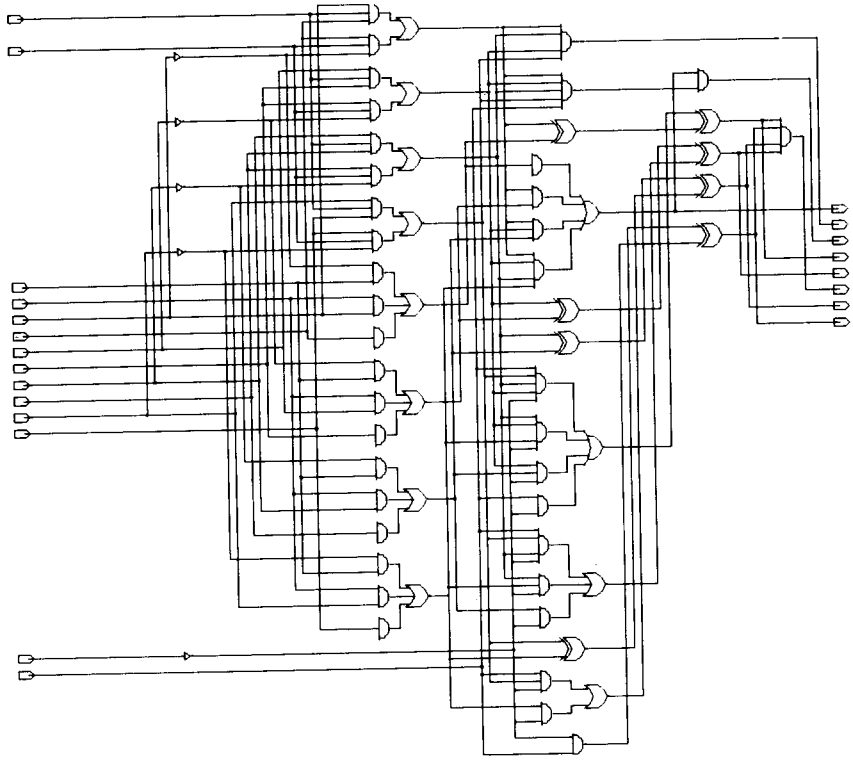

Fig. 12. SN74181 arithmetic logic unit chip.

Table 1
Test cases statistics and run time

Circuit name	Number of modules	Number of connections	Run time * (s.)
J K master slave F.F.	11	21	4.27
Binary counter	28	69	14.37
SN74181	32	126	19.86

* Time unit: SUN 3/110 CPU SEC

5. Conclusions

In this paper, we have presented a systematic heuristic placement and routing method to consider the automatic schematic generation in detail. The features of the method are summarized as follows:

(i) The levelling algorithm partitions the circuit from left to right and processes the feedback loops.
(ii) The bubbling algorithm resolutes the circuit to benefit the initial placement, trunk routing, and pin assignment.
(iii) The trunk routing algorithm controls the fanout and feedback routing in human style.
(iv) The level expansion benefits the normal distribution in placement.
(v) The channel size estimation and track assignment benefit the compact and regular routing.

By combining the above features with heuristics, the ASG can efficiently generate schematic diagrams which are very close to the drawings of human designers. The examples show that the proposed system works perfectly.

References

[1] Kumar, A., Arya, A., Swaminathan, V.V. and Misra, A., Automatic generation of digital system schematic diagrams, *IEEE Des. Test Comput.* **2** (1986) 58–65.
[2] Kurdahi, F.J. and Parker, A.C., REAL: A Problem for REgister ALlocation, *Proc. 24nd Design Automation Conference* (1987) 203–215.
[3] Chen, G.D. and Parng, T.M., A database management system for a VLSI design system, *Proc. 25nd Design Automation Conference* (1988) 257–262.
[4] Stok, L. and Koster, G.J.P., From network to artwork, *Proc. 26nd Design Automation Conference* (1989) 686–689.
[5] Majewski, M.A., Krull, F.N., Fuhrman, T.E. and Ainslie, P.J., AUTODRAFT: automatic synthesis of circuit schematics, *Proc. ICCAD* (1986) 435–438.
[6] Ahlstrom, M.L., Hadden, G.D. and Stroick, G.R., An expert system for the generation of schematics, *Proc. ICCAD* (1984) 720–725.
[7] Chun, R.K., Chang, K.J. and McNamee, L.P., VISION: VHDL induced schematic imagine on net-lists, *Proc. 24nd Design Automation Conference* (1987) 436–442.